2007 *ВЕСТНИК Санкт-Петербургского университета Сер. 10* **Вып. 4**

ИНФОРМАТИКА

УДК 004.434:004.42

Романовский К.Ю, Кознов Д.В.

ЯЗЫК DRL ДЛЯ ПРОЕКТИРОВАНИЯ И РАЗРАБОТКИ ДОКУМЕНТАЦИИ СЕМЕЙСТВ ПРОГРАММНЫХ ПРОДУКТОВ

Введение. Семейство программных продуктов (product line) — это набор продуктов, совместно продвигаемых на рынке и разрабатываемых на основе общих активов с помощью хорошо определенного метода повтороного использования [1]. Идея разрабатывать не один продукт, а сразу целое семейство была предложена в 1976 г. Парнасом [2]. К настоящему времени эта идея нашла широкое практическое воплощение. Сформировалось два научно-практических центра, которые анализируют и распространяют успешный индустриальный опыт в этой области, создают стандарты, организуют конференции и т.д. Это Институт Программной Инженерии[1] университета Карнеги-Мелон в США, и Европейский Институт Программного Обеспечения[2].

В качестве повторно используемых активов выступают программные компоненты, модели анализа и проектирования, практики менеджмента, тесты и пр. [1]. Однако документация в этот перечень почему-то не попала, а различные концепции, методы и подходы к проектированию и разработке семейств программных продуктов (обзоры можно найти в работах [1, 3, 4]) обходят задачу организации повторного использования документации стороной. В отдельных отчетах об индустриальных внедрениях product line подхода упоминается о повторном использовании документации (например, в [5]), но на этом не делается акцента.

А между тем задача создания документации при разработке ПО является важной и трудоемкой. Объем пользовательской и проектной документации часто оказывается весьма значительным, ее требуется оперативно изменять, необходимы различные представления документации — компьютерные справочные системы, документы–книги, Интернет–документация и т.д. Существуют многочисленные подходы и технологии разработки технической документации — такие известные продукты как Adobe FrameMaker [6], AuthorIT [7], различные закрытые и открытые XML-форматы для представления документации — DocBook [8], DITA [9], Juniper DTD [10] и др. В некоторых из них, в частности, в DocBook и DITA, нашла отражение задача повторного использования. Однако, эта задача решается там как вспомогательная, не затрагиваются такие важные аспекты, как проектирование сложных пакетов документации с учетом повторного использования, адаптация повторно используемых фрагментов к требованиям контекста использования и др.

Продолжая развивать метод разработки документации для семейств программных продуктов, предложенный нами в [11], в данной работе мы предлагаем новый язык DRL (Documentation Reuse Language) для проектирования и разработки документации

[1]SEI — Software Engineering Institute, http://www.sei.cmu.edu/plp.

[2]ESI - European Software Institute, http://www.esi.es.

с акцентом на повторное использование. Этот язык охватывает все этапы разработки сложной документации - от проектирования до публикации электронных документов. В DRL для проектирования структуры документации с учетом повторного использования включены графические средства, основанные на диаграммах возможностей (Feature Diagrams) [12, 13, 14]. Собственно разработку документации предлагается выполняется на основе специального XML-формата, также входящего в состав DRL, определенного в терминах XML Schema. Наконец, для определения полиграфических свойств документов (глав, разделов, форматирования) в DRL допускаются вставки на языке DocBook. Полная DRL- спецификация транслируется в DocBook, что позволяет использовать средства среды DocBook для автоматической генерации целевых документов в различных форматах (PDF для печатных документов, HTML для электронных документов и пр.). Описание DRL строго формализовано: определены абстрактный синтаксис (с помощью НФБН-грамматики [3]), графическая нотация (с помощью графического реасширения грамматик НФБН[4]) и сериализационный синтаксис (в виде XML Schema[5]).

1. Обзор существующих подходов. Данная работа находится на стыке двух предметных областей: (а) методов создания семейств программных продуктов и (б) технологий и языков разработки технической документации. В первой области выделим следующие подходы, которые мы использовали в нашей работе:

- диаграммы возможностей (Feature Diagrams) [12, 13, 14], предназначенные для визуального отображения общих и частных свойств набора продуктов;
- язык XVCL [15], основанный на подходе Пола Бассета [17, 16] и предназначенный для языково-независимого управления повторным использованием программного кода.

Среди технологий и языков разработки технической документации[6] существуют следующие подходы и формализмы, которые сходны по задачам с DRL:

- технология DocBook, предназначенная для полиграфического форматирования текстов и последующей публикации их в различных выходных представлениях — HTML, PDF и т.п.;
- подход DITA, предназначенный для разработки сложной документации и поддерживающий крупноблочное повторное использование фрагментов текста.

Рассмотрим эти подходы более подробно.

1.1. Диаграммы возможностей (Feature Diagrams) предложены в 1990 г. в рамках метода FODA [13] группой исследователей из института программной инженерии США во главе с Кио Кангом (Kyo Kang). В настоящий момент имеется ряд инструментальных пакетов, поддерживающих создание диаграмм возможностей.

[3]НФБН (Нормальная Форма Бэкуса-Наура) — язык описания грамматик, является стандартом дефакто при формализации абстрактного синтаксиса языков программирования

[4]Использовался формализм, налагочиный тому, который применен для описания языка SDL/GR [21].

[5]XML Schema — международный стандарт описания типов XML-документов (http://www.w3.org/XML/Schema).

[6]Обзор таких технологий с акцентом на средствах повторного использования представлен нами в работе [11].

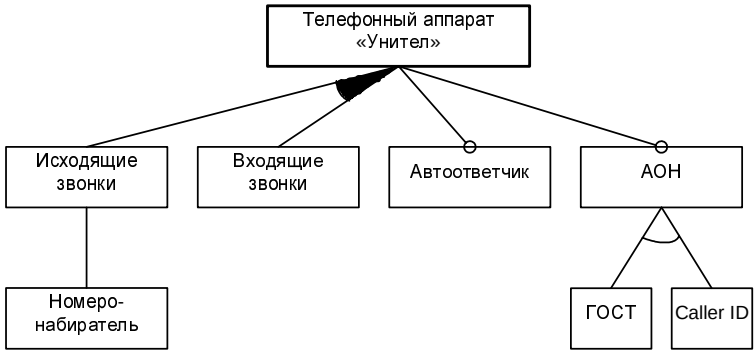

Рис. 1: Диаграмма возможностей.

Основная задача диаграмм возможностей — формализовать общие и различные черты систем, входящих в семейство продуктов. Ключевым понятием формализма является возможность (feature) — обособленное свойство системы, распознаваемое с точки зрения пользователя или разработчика. Таким образом, диаграмма возможностей является набором возможностей и иерархических связей между ними с явно выделенным корнем иерархии, который называется концептом (concept) [12]. Иерархическая связь отображает декомпозицию концепта или возможности и называется отношением включения. На диаграмме могут быть предусмотрены точки вариации - необязательные включения, а также условия на группы включений, которые могут по-разному интерпретироваться в контексте различных систем, описываемых диаграммой. Пример диаграммы возможностей показан на Рис. 1. На этом рисунке концепт — Телефонный аппарат “Унител” — символизирует семейство телефонных аппаратов, поддерживающих различные функции. В целом такие аппараты во многом похожи, однако есть и различия. На данной диаграмме показано, что телефонные аппараты должны поддерживать исходящие звонки или входящие звонки (или и те и другие), могут содержать автоответчик и АОН (автоматический определитель номера), который, в свою очередь, реализуется одним из двух способов — либо в российском стандарте ГОСТ, либо в международном Caller ID.

Диаграммы возможностей в оригинальном варианте могут использоваться только на этапах анализа и, в меньшей степени, проектирования ПО и никак не связаны с артефактами последующих этапов, такими как программный код и документация.

1.2. Язык XVCL и подход Бассета. XML-based Variant Configuration Language (XVCL) — язык конфигурирования вариантов, основанный на XML, предложен Станиславом Джарзабеком и группой ученых из университета города Сингапур в 2000 г. [15]. В его основе лежит подход Пола Бассета [17, 16], предложенный в 1987 г. и успешно использованный на практике для реструктуризации и реинжиниринга COBOL-приложений в составе продукта Netron Fusion [18].

Язык XVCL является макроязыком, построенным на технологии XML. Он предназначен для разметки текстов на языках программирования с целью выделения повторно используемых фрагментов, не совпадающих с языковыми конструкциями — процедурами, классами и пр. Такие фрагменты в терминологии XVCL называются фреймами. Основное отличие XVCL от других подходов к повторному использованию

— предоставление возможностей модифицировать содержание фрейма в конкретной точке (так называемые точки расширения) из того контекста, в котором фрейм используется.

Несмотря на интересные идеи, заложенные в язык XVCL, он слишком абстрактным. Для успешного практического использования XVCL нуждается в серьезных модификациях для адоптации к целевой предметной области. Так, в нашем случае необходима не просто фрейм-разметка целевых спецификаций документации, а реализация специальных языковых конструкций, позволяющих моделировать структуру документации, различные отношения между фрагментами документов, а также специальные способы повторного использования, которые характерны именно для области разработки документов.

1.3. Технология DocBook была впервые предложена в 1991 г. компаниями HaL Computer Systems и O'Reilly & Associates. На сегодняшний день формат DocBook стандартизован консорциумом OASIS[7] и поддержан множеством технологических средств — пакетами FrameMaker, XMLSpy[8], Oxygen[9], VEX[10] и многими другими. Формат DocBook специфицирован в различных XML-форматах: DTD, XML Schema или Relax NG. DocBook является открытым стандартом, имеются также открытые реализации преобразований, обеспечивающих публикацию документов в различные целевые форматы. В настоящее время DocBook широко используется для разработки документации операционных систем семейства UNIX (FreeBSD, Linux), а также в мире разработки открытого ПО.

Язык DocBook является XML-языком, содержащим конструкции для определения полиграфической структуры документов (например, части, главы, параграфы) и для высокоуровневого форматирования текстов (например, выделение текста, вставка иллюстраций). В DocBook разделяется детальное описание форматирования и описание содержимого публикации. Так, например, есть конструкция <book /> для определения списка авторов, названия документа и разнообразной дополнительной информации о документе. Конкретное представление этой конструкции определяется в наборе преобразований в целевые форматы. При этом в различных целевых форматах представление может отличаться, что является воплощением идеи единого исходного представления (single source) [19] — возможность на основе одного исходного документа с помощью набора преобразований получать целевые документы различного формата (PDF, WinHelp, HTML и др.).

С точки зрения повторного использования, DocBook, фактически, предлагает лишь реализацию идеи единого исходного представления. При этом, не учитывая, что документы разного формата, как правило, и структурированы по-разному [20]. Также отсутствуют встроенные средства поддержки разделения документации на части, и для этого приходится пользоваться штатными средствами XML, что существенно усложняет поддержание ссылочной целостности.

1.4. Подход DITA (Darwin Information Typing Architecture — архитектура типизации информации Дарвина) предложен компанией IBM в 2001 г. для разработки

[7]OASIS - Organization for the Advancement of Structured Information Standards) - международный некоммерческий консорциум, занимающийся разработкой и внедрением стандартов в области электронной коммерции. http://www.oasis-open.org.

[8]XMLSpy — пакет разработки XML-документов от компании Altova http://www.altova.com/.

[9]Oxygen — пакет разработки XML-докуемнтов от компании SyncRO Soft Ltd, интегрируемый в платформу Eclipse, http://www.oxygenxml.com/.

[10]VEX (Visual Editor for XML) — XML-редактор с открытым исходным кодом, интегрируемый в платформу Eclipse, http://vex.sourceforge.net/.

сложной модульной технической документации. Позднее DITA стандартизован консорциумом OASIS. Данный подход поддерживается рядом известных инструментальных средств, таких как XML Spy, Oxygen и др.

Подход DITA содержится XML-язык для спецификации документации. Его ключевой конструкцией является топик (topic) — самостоятельный фрагмент документации, который может использоваться в различных контекстах. DITA предлагает механизм типизации топиков, предполагающий предварительную разработку шаблона топика и последующее массовое создание топиков по шаблону. В стандарте предлагается 3 типа топиков — понятие (описание какого-либо понятия, например класса в ООП), задача (последовательность шагов для реализации какого-либо действия, например для подключения устройства к компьютеру) и справочник (подробное описание какого-либо объекта, например команды пользовательского интерфейса), каждый из которых имеет свою структуру. Наряду с набором предопределенных типов, имеется механизм создания собственных типов топиков.

Топики обеспечивают крупноблочное повторное использование. Также DITA поддерживает простейшее "мелкозернистое" повторное использование с помощью механизма ссылок на содержание: любой тег может быть заменен на тег аналогичного типа с указанным идентификатором.

Основное преимущество DITA по сравнению с DocBook — принцип модульности документации. Однако DITA, так же как и DocBook, не предлагает механизмов настройки переиспользуемых элементов по требованиям контекста использования, которыми обладает, например, язык XVCL.

1.5. Выводы обзора. Формализмы, используемые при проектировании и организации повторного использования различных активов, не рассчитаны на документацию, а языки и средства разработки документации не поддерживают специфичных для семейств программных продуктов потребностей. В различных подходах к разработке документации реализуются лишь отдельные идеи: единое исходное представление в DocBook [8], модульность в DITA [9], адаптируемость во фреймах Бассета [16] и языке XVSL [**?**]. Однако отсутствует единый язык, ориентированный на разработку документации семейств программных продуктов и охватывающих весь процесс разработки от анализа и проектирования документов до их публикации.

2. Описание языка. В языке DRL, подобно SDL[11] [21], предусмотрено две нотации — графическая и текстовая. Графическая нотация ориентирована на проектирование структуры документации, а также полезна в ситуациях, когда необходимо быстро получить обзорную информацию о документации в целом или о каком-либо ее фрагменте. Все, что можно выразить с помощью графической нотации, можно выразить также и в текстовом виде. Тестовое представление, таким образом, шире графического.

2.1. Графические средства проектирования. DRL поддерживает три вида диаграмм:

- главная диаграмма,
- диаграмма вариативности,

[11] Specification and Description Language - язык для моделирования телекоммуникационных систем, развиваемый с 1976 года международным комитетом CCITT, переименованным позднее в ITU-T (http://www.itu.int/ITU-T/). Данный язык охватывает полный цикл разработки телекоммуникационного ПО, включая статическую и динамическую составляющие систем. SDL предлагает две различные нотации — графическую для наглядного представления алгоритмов и текстовую, предназначенную для подробной спецификации.

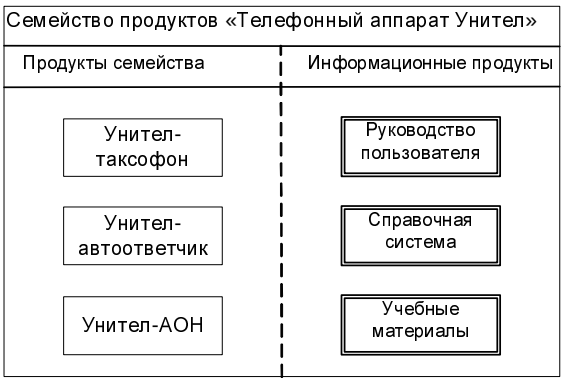

Рис. 2: Главная диаграмма.

- диаграмма продукта.

Главная диаграмма позволяет определить все продукты семейства и все главные шаблоны документов, а также связать продукты с шаблонами. Пример главной диаграммы показан на Рис.2. На этой диаграмме представлено семейство продуктов "Телефонный аппарат Унител". Это программно-аппаратное семейство содержит ряд телефонных аппаратов различного назначения и функциональности — начиная от таксофонов и заканчивая полнофункциональными системами с автоответчиком и встроенным определителем номера. Пакеты документов различных продуктов семейства очень похожи, но, тем не менее, различаются. Как минимум в документации ряда продуктов отсутствует описание функций, в них отсутствующих (например, в документации "Унител-таксофон" отсутствует описание модуля входящих звонков, поскольку таксофон не предоставляет такой услуги).

Семейство состоит из трех продуктов — "Унител-таксофон", "Унител-автоответчик", "Унител-АОН". Продукты семейства показываются в левой секции диаграммы, а в правой представлены шаблоны различных целевых пользовательских документов: "Руководство пользователя" (как правило, единый документ в формате PDF), "Справочная система" (набор разделов справки, как правило в формате WinHelp или HTMLHelp), "Учебные материалы" (достаточно часто представлены в виде HTML-страниц). Это именно шаблоны, называемые в терминологии DRL *информационными продуктами* (ИП) и содержащие наборы глав и частей и пр. ИП настраивается и конкретизируется для каждого продукта семейства, превращаясь в документ.

Каждый ИП представляется в DRL с помощью *диаграмм вариативности*, основанных на диаграммах возможностей. Пример такой диаграммы для ИП "Руководство пользователя" представлен на Рис.3. ИП находится в корне иерархии и изображается в виде прямоугольника с двойными границами. Остальные узлы данного дерева — это *информационные элементы* (ИЭ): блоки документации, подготовленные для повторного использования. ИЭ могут состоять из других ИЭ. Эти отношения могут иметь различную семантику: строгое альтернативное включение (XOR), нестрогое альтернативное включение (OR-включение), обязательное и необязательное включение.

Рассмотрим пример. На Рис.3 представлен ИП "Руководство пользователя", который состоит из следующих ИЭ: "Документация модуля "Исходящие звонки", "Докумен-

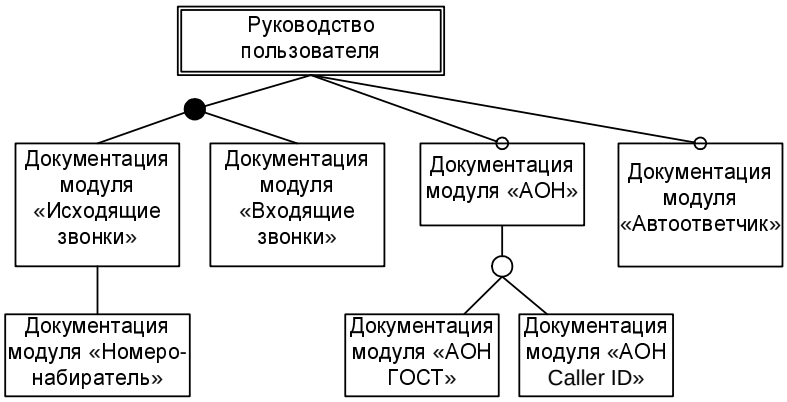

Рис. 3: Диаграмма вариативности.

тация модуля “Входящие звонки”, “Документация модуля “Автоответчик” и “Документация модуля “АОН”. Первый и последний ИЭ также, в свою очередь, декомпозируются. Связь между ИП “Руководство пользователя” и ИЭ “Документация модуля АОН” является необязательной — то есть для некоторых продуктов семейства “Документация модуля “АОН” может быть опущена. Связи между ИП “Руководство пользователя” и ИЭ “Документация модуля “Исходящие звонки” и “Документация модуля “Входящие звонки” объединены в OR-группу (изображена черным кружком, из которого связи расходятся). Это означает, что в руководство пользователя конкретного продукта может быть любой непустой набор ИЭ, соответствующих входящим в группу связям. Аналогично объединены в группу (только уже в XOR-группу) связи между ИЭ “Документация модуля “АОН ГОСТ” и “Документация модуля “АОН Caller ID”. В документацию конкретного продукта может войти один и только один ИЭ из этой группы.

Для описания документации конкретных продуктов служит следующий тип диаграмм — *диаграмма продуктов*. На Рис.4 представлен пример таких диаграмм для руководства пользователя Унител-таксофона (а) и Унител-автоответчика (б). На этих диаграммах, в отличие от предыдущего типа диаграмм, запрещены любые неопределенности, т.е. все включения должны быть обязательными и не должны быть групп включений. Диаграмма документации продукта получается из диаграммы информационного продукта путем разрешения всех точек неопределенности, т.е. выбором нужных ссылок в группах ссылок (по правилам соответствующих групп) и удалением ненужных ссылок из числа необязательных.

Далее документацию семейства нужно детализировать с помощью текстовой нотации DRL.

2.2. Введение в текстовый DRL. Диаграмма с Рис. 2 транслируется в текстовое DRL-описание. Листинг 1 иллюстрирует текстовое представления продуктов семейства. Конструкция *ProductLine* обозначает семейство продуктов, конструкция *PLScheme* - схему семейства продуктов, конструкция *Product* указывает на отдельный программный продукт семейства.

Шаблоны документов семейства (информационные продукты) в нашем случае транслируются в DRL-текст, показанный в Листинге 3. Конструкция *PLDocumentation* обо-

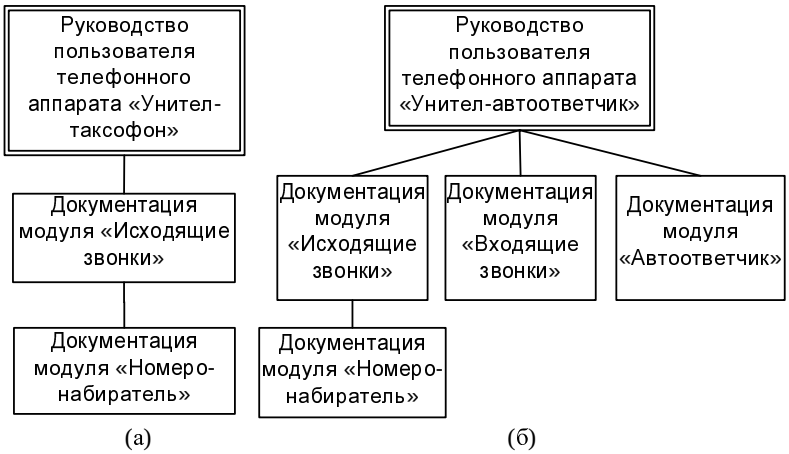

Рис. 4: Пример диаграмм продукта.

```
<ProductLine name='Семейство телефонных аппаратов "Унител"'>
  <PLScheme>
    <Product name="Унител-таксофон" id="tax"/>
    <Product name="Унител-автоответчик" id="answer"/>
    <Product name="Унител-АОН" id="callerid"/>
  </PLScheme>
  <PLDocumentation/>
</ProductLine>
```

Листинг 1: Пример описания семейства продуктов.

значает документацию семейства, конструкция *DocumentationCore* - это ядро документации, т.е. та ее часть, которая содержит набор повторно используемых компонент. Конструкция *InfProduct* указывает на ИП. Конструкция *InfElement* обозначает ИЭ.

Соответствующая диаграмме с Рис. 3 текстовая DRL-спецификация показана в Листинге 3, который отражает часть диаграммы — корневой элемент (ИП) и узлы первого уровня (ссылки на ИЭ). Группа ИЭ определяется отдельной конструкцией *InfElemRefGroup*. Листинг 3 также иллюстрирует использование кода DocBook внутри элементов DRL.

2.3. Крупноблочное повторное использование. В большинстве случаев, повторное использование эффективно, только если возможна адаптация компонент к требованиям конкретного контекста использования (вариативность). В DRL поддерживаются несколько механизмов вариативности. Наряду с простыми механизмами, такими как псевдопеременные и условные блоки, используются также и точки расширения. Точки расширения задаются конструкцией *Nest*. Пример ИЭ с точками расширения показан в Листинге 4. В данном примере точки расширения созданы для того чтобы можно было добавить или удалить упоминание тех или иных способов индикации номера звонящего абонента в зависимости от особенностей конкретного телефонного аппарата (голосовая индикация, визуальная индикация, поиск контакта в адресной книге и т.п.). Для реального использования документации семейства необходимо описать, как

```
<PLDocumentation>
  <DocumentationCore>
    <InfProduct name="Руководство пользователя" id="userguide">:
    </InfProduct>
    <InfProduct name="Справочная система" id="String">:</InfProduct>
    <InfProduct name="Учебные материалы" id="String">:</InfProduct>
    <InfElement name="АОН" id="АОН">:</InfElement >
  </DocumentationCore>
</PLDocumentation>
```

Листинг 2: Пример описания шаблонов документации.

```
<InfProduct name="Руководство пользователя" id="userguide">
    <book>
      <bookinfo>
      <title><dictref dict="main" Entry="productname"/></title>
      </bookinfo>
      <InfElemRefGroup id="in_out" relation="OR" />
      <InfElemRef id="out_ref" infelem="Исходящие" groupid="in_out"/>
      <InfElemRef id="in_ref" infelem="Входящие" groupid="in_out"/>
      <InfElemRef id="answer_ref" infelem="Автоответчик" optional="true"/>
      <InfElemRef id="aon_ref" infelem="АОН" optional="true"/>
    </book>
</InfProduct>
```

Листинг 3: Пример описания информационного продукта.

именно документация специализируется для каждого продукта семейства. Разработка документации продукта начинается с создания специализированных информационных продуктов (*FinalInfProduct*). Специализированный информационный продукт (СИП) - это реализация ИП из документации семейства, привязанная к конкретному продукту семейства. Пример описания СИП показан в Листинге 5. СИП описывает набор информационных элементов с единственным допустимым видом отношений - обязательное включение. Выбор и настройка информационных элементов осуществляется с помощью адаптеров (*Adapter*). Адаптер описывает настройки, которые необходимо применить к той или иной ссылке на информационный элемент в контексте заданного продукта семейства. Пример описания адаптера показан в Листинге 5.

2.4. "Мелкозернистое" повторное использование. При разработке технической документации помимо "крупноблочного" повторного использования нередко требуется организовать повторное использование отдельных слов или фрагментов предложений. В терминологии DRL подобное повторное использование называется "мелкозернистым". В частности, при мелкозернистом повторном использовании возникает задача адаптации словоформ (падежа, числа и т.п.) к особенностям контекста использования.

Для организации "мелкозернистого" повторного использования предназначены словари и каталоги. Словарь — это набор пар имя-значение. Пример описания словаря иллюстрирует Листинг 6.

На практике значение некоторых элементов словаря может зависеть от конкретного продукта - например для таких элементов, как название продукта и версия продукта,

```
<InfElement id="АОН CallerID" name="АОН CallerID">
  <Nest id="Индикация номера" descr="Индикация номера">
    Индикация номера осуществляется следующим способом.
  </Nest>
  <Nest id="Способы индикации номера">
    При успешном определении номер отображается на
    встроенном дисплее телефонного аппарата.
  </Nest>
</InfElement>
```

Листинг 4: Пример описания информационного элемента с точкой расширения.

```
<ProductDocumentation productid="callerid">
  <FinalInfProduct infproductid="userguide">
    <Adapter infelemrefid="CallerID_aon_ref">
      <Insert-After nestid="Способы индикации номера">
        Также предусмотрена возможность звуковой индикации номера абонента.
      </Insert-After>
    </Adapter>
  </FinalInfProduct>
</ProductDocumentation>
```

Листинг 5: Пример описания информационного продукта и адаптера.

которые встречаются в документации любой системы. В DRL для реализации продуктозависимых словарей предусмотрена возможность размещения словаря внутри документации конкретного продукта. При поиске элементов словаря сначала осуществляется просмотр словаря с заданным идентификатором в данном продукте, а если такого словаря нет, либо в нем отсутствует требуемое значение, подключается словарь из ядра документации.

Помимо словарей DRL предлагает также конструкцию *Directory* (каталог). Каталог содержит набор кортежей имя-набор атрибутов. Распространенный пример каталога — команды пользовательского интерфейса приложения. Для описания команд применяется большое количество различных данных — название команды, подсказка, пиктограмма, горячая клавиша и т.п. Тем не менее, в документации упоминания команды могут отличаться по подробности. При описании панели инструментов, например, осмысленно использовать название команды и пиктограмму, а в справочнике команд - напротив, все информационные поля. Для реализации различных способов отображения элемента каталога используется конструкция *DirTemplate* (шаблон отображения элемента каталога). Листинг 7 иллюстрирует описание каталога (*Directory*), шаблона отображения (*DirTemplate*) и использования элемента каталога (*DirRef*). Каталоги, так же как и словари, могут быть продукто-зависимыми.

3. Формальные средства спецификации языка. Для формального описания языка DRL используется метод, описанный в [3]. Там классическое описание языка (синтаксис, семантика, прагматика) уточняется в части описания синтаксиса, разбиваясь на абстрактный, конкретный и сериализационный синтаксис. Абстрактный синтаксис в общем виде описывает типы языковых конструкций и правила их сочетания. Конкретный синтаксис определяет, как выглядят конструкции языка при написании

```
<Dictionary id="main">
  <Entry id="productname">Название продукта</entry>
  <Entry id="productversion">1.0</entry>
</Dictionary>
```

Листинг 6: Пример описания словаря.

```
<Directory id="commands">
  <Entry id="About">
    <Attr id="name">О программе</Attr>
    <Attr id="hint">Информация о программе</Attr>
    <Attr id="icon">about.jpg</Attr>
  </Entry>
</Directory>
<DirTemplate id="CommandNameHint" directoryid="commands">
  <AttrRef attrid="name"></AttrRef>  (<AttrRef attrid="hint"></AttrRef>)
</DirTemplate>
<InfElement name="Command List" id="CommandList">
 Команды: <DirRef templid="CommandNameHint" entryid="About"/>, :
</InfElement>
```

Листинг 7: Пример описания каталога, шаблона отображения элемента каталога и включения элемента каталога.

текстов. Сериализационный синтаксис определяет форму представления конструкций языка, пригодную для долгосрочного хранения и передачи. Кроме этого мы добавили таблицы соответствия графических и текстовых конструкций языка.

В нашем случае абстрактный синтаксис позволяет объединить графическую и текстовую части языка в единое описание. Конкретный синтаксис для DRL — это описание графической нотации. Сериализационный синтаксис совпадает у нас с конкретным синтаксисом текстовой части языка, поскольку текстовый DRL является XML-языком.

Для описания абстрактного синтаксиса мы использовали форму представления грамматик НФБН. Пример спецификации абстрактного синтаксиса конструкции "Документация семейства" показан в Листинге 8.

Для описания текстовой нотации DRL используется язык XML Schema, являющийся промышленным стандартом для схем документов XML. Все конструкции DRL содержатся в пространстве имен "drl".

Для описания графической нотации используется расширение НФБН. В исходную грамматику добавлены следующие бинарные инфиксные операторы: *содержит, соединяется с*. Так, например, набор правил на Рис. 5 (а) описывает конструкцию, изображенную на Рис. 5 (б).

Не всегда элементы графической нотации очевидным образом отображаются на элементы текстовой нотации. Спецификация DRL содержит таблицы соответствия абстрактного синтаксиса и конкретной нотации. Фрагмент такой таблицы соответствия абстрактного синтаксиса и текстовой нотации показан в Табл. 1.

Представленное формальное описание языка позволяет однозначно зафиксировать весь набор его конструкций. Совместно с открытым исходным кодом инструментария поддержки языка, такая спецификация обеспечивает широкие возможности расшире-

```
<Документация семейства> :: <Информационный продукт>+
  <Информационный элемент>*
  <Словарь>*
  {<Каталог> <Шаблон элемента каталога>+ }*
```

Листинг 8: Абстрактный синтаксис: конструкция “Документация семейства”.

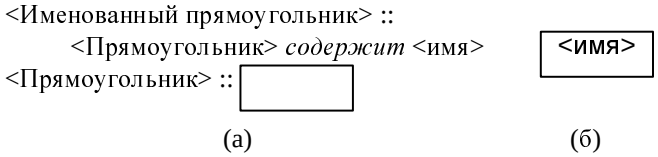

Рис. 5: Пример описания графической нотации.

ния языка как в промышленных условиях применения (как описано в работе [22]), так и в исследовательских целях. Отсутствие подобной полной спецификации, например, для языка моделирования веб-приложений WebML [23] очень сильно затрудняет расширение языка, развитие и создание для него новых программных инструментов.

4. Сравнения и соотнесения. Ряд принципов языка DRL основывается на хорошо зарекомендовавших себя идеях существующих языков. Диаграммы вариативности DRL основывается на диаграммах возможностей [12]. Для удобства добавления и удаления включения информационного элемента в (из) группу мы добавили в нотацию диаграмм возможностей новый тип узла для обозначения типа включения.

Некоторые вспомогательные конструкции DRL основаны на конструкциях инструментального пакета FrameMaker [6]. Конструкции форматирования текста позаимствованы из языка DocBook [8], оттуда же взята идея единого исходного представления текста.

Новыми являются конструкции, обеспечивающие “мелкозернистое” переиспользование текста, а именно конструкции определения и работы с каталогами. Также новым является понятие адаптера, разработанное на основе конструкции COPY фреймов Бассета [16] и конструкции adapt технологии XVCL [15]. Основное отличие адаптера DRL от указанных конструкций — это возможность разделения описания семейства документов и специализации этих документов под требования конкретного продук-

Абстрактный синтаксис	Текстовая нотация

<Модификатор группы>	`<xs:simpleType name="GroupModifier">`
	`  <xs:restriction base="xs:string">`
	`  <xs:enumeration value="OR"/>`
	`  <xs:enumeration value="XOR"/>`
	`  </xs:restriction>`
	`</xs:simpleType>`

Таблица 1: Таблица соответствия абстрактного синтаксиса и текстовой нотации.

та. Оригинальные конструкции COPY и adapt одновременно подключают и специализируют переиспользуемый фрагмент. В языке DRL подключение переиспользуемого фрагмента осуществляется отдельной конструкцией ("включение информационного элемента"), поскольку возможность подключения определяется на этапе проектирования документации семейства без обязательной привязки к контексту того или иного продукта. Специализировать переиспользуемый фрагмент можно с помощью адаптера непосредственно в контексте разработки документации конкретного продукта, не затрагивая документацию семейства.

5. Заключение. Для DRL была выполнена пилотная реализация в среде Microsoft .NET/Visio. Подход был апробирован в проекте разработки документации семейства систем автоматизации телевещания, о чем рассказано в работе [22]. В результате использования DRL был достигнут высокий уровень повторного использования текста.

В настоящий момент происходит реализация DRL на платформе Eclipse/GMF, с поддержанием полного цикла разработки документации и созданием программных средств, облегчающих техническим писателям работу с DRL.

Также перспективным представляется добавление в DRL дополнительных средств для пакетизации: подсемейств, пакетов информационных продуктов, пакетов информационных элементов.

Список литературы

1. *Clements, P., Northrop, L.* Software Product Lines: Practices and Patterns. Boston, MA: Addison-Wesley, 2002. 608 p.

2. *D. Parnas.* On the Design and Development of Program Families. // IEEE Transactions on Software Engineering, March 1976.

3. *Greenfield J., Short K., Cook S., Kent S.* Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. Indianapolis, Indiana: Wiley Publishing, Inc., 2004. 696 p.

4. *Northrop L.* SEI's Software Product Line Tenets. // IEEE Software July–August 2002. P. 32–40

5. *Jaaksi A.* Developing Mobile Browsers in a Product Line. // IEEE Software, July–August 2002, P. 73–80.

6. *Marques. M.* Single-sourcing with FrameMaker. // TECHWR-L Magazine Online, http://www.techwr—l.com/techwhirl/magazine/technical/singlesourcing.html

7. *James-Tanny C.* Single-sourcing with AuthorIT. // Single—Sourcing SIG online newsletter. June 2002. http://www.stcsig.org/ss/articles062002/6_02_ss_authorit.htm

8. *Walsh N., Muellner L.* DocBook: The Definitive Guide. O'Reilly, 1999. 644 p.

9. *Day, D., Priestley, M., Schell, David A.* Introduction to the Darwin Information Typing Architecture - Toward portable technical information. // http://www—106.ibm.com/developerworks/xml/library/x—dita1/

10. *Fraley L.* Beyond theory: making single—sourcing actually work. // Proceedings of the 21st annual international conference on Documentation. ACM Press New York. 2003. P. 52–59

11. *Романовский К.Ю.* Метод разработки документации семейств программных продуктов. // Системное программирование. Вып. 2. Сб. статей / Под ред. А.Н.Терехова, Д.Ю.Булычева.. СПб.: Изд-во СПбГУ, Готовиться к печати в 2007 г.

12. *Czarnecki K., Eisenecker U.* Generative Programming: Methods, Tools, and Applications. Reading, Mass.: Addison Wesley Longman, 2000. 864 p.

13. *Kang K., Cohen S., Hess J., Novak J., et al.* Feature-Oriented Domain Analysis (FODA) Feasibility Study. // Technical Report, CMU/SEI—90—TR—21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1990

14. *Kang K., Jaejoon L, Donohoe P.* Feature-Oriented Product Line Engineering. // IEEE Software July/August 2002. - P. 58–65.

15. *Jarzabek, S., Bassett, P.* Hongyu Zhang, Weishan Zhang. XVCL: XML-based variant configuration language. // Proceedings of 25th International Conference on Software Engineering, 2003. P. 810–811.

16. *Bassett P.* Framing software reuse - lessons from real world. Prentice Hall, 1996. 365 p.

17. *Bassett P.* Frame-Based Software Engineering. // IEEE Software. Volume 4. 1987. Number 4. P. 9–16.

18. *Netron* Fusion // http://www.netron.com/

19. *Rockley A., Kostur P., Manning S.* Managing Enterprise Content: A Unified Content Strategy. New Riders, 2002. 592 p.

20. *Clark D.* Rhetoric of present single-sourcing methodologies. // Proceedings of the 20th annual international conference on Computer documentation. ACM Press, New York. 2002. P. 20–25

21. *ITU—T* Recommendation Z.100: Specification and description language (SDL). 1999. 244 p.

22. *Кознов Д.В., Перегудов А.Ф., Романовский К.Ю., Кашин А, Тимофеев А.* Опыт использования UML при создании технической документации. // Системное программирование. Вып. 1. Сб. статей / Под ред. А.Н.Терехова, Д.Ю.Булычева.. СПб.: Изд-во СПбГУ, 2005. С. 18–36.

23. *Ceri S., Fraternali P., Bongio A.* Web Modeling Language (WebML): a modeling language for designing Web sites. // Computer Networks, Volume 33, Numbers 1–6. 2000. P. 137–157

Статья приянта к печати xx xxxxxxxxx 2007 г.

УДК 004.434:004.42

Романовский К.Ю, Кознов Д.В. Язык DRL для проектирования и разработки документации семейств программных продуктов. // Вестн. С.-Петербург. ун-та. Сер. 10. 2007. Вып. 4. С. 00–00

Предложен язык DRL для проектирования и разработки документации семейств программных продуктов. Язык имеет визуальное представление для высокоуровневого проектирования структуры сложной документации и текстовое представление для гибкого управления повторным использованием и описания форматирования текстов. Представленный язык сравнивается с существующими технологиями и подходами разработки сложной документации.

Библиогр. 23 назв. Ил. 5. Табл. 1.

Summary

Romanovsky K.Y., Koznov D.V. DRL language for design and development of software product line documentation.

In this paper the DRL language is presented. This language is intended for design and development of software product lines documentation. The language has visual and textual representations, it implements Feature Diagrams to improve a design of a large-scale document reuse and provides means for fine-grained reuse of small text parts. Reuse management mechanism permits adaptation of both large and small reusable parts to peculiarities of usage context. Text notation of DRL covers all stages of complex documentation development starting from design and up to electronic document publication. Textual part of DRL is integrated with popular documentation markup language DocBook to improve publication features.

Романовский К.Ю, Кознов Д.В. Язык DRL для проектирования и разработки документации семейств программных продуктов.

В данной работе предлагается язык DRL (Documentation Reuse Language). Этот язык предназначен для проектирования и разработки документации семейств программных продуктов. Язык имеет визуальное и текстовое представление, реализует Feature Diagrams для удобства проектирования повторного использования крупных фрагментов документов, а также предлагает средства для “мелкозернистого” повторного использования небольших фрагментов текста. Механизм управления повторным использованием допускает адаптацию к особенностям контекста использования как крупных фрагментов, так и “мелкозернистых”. Текстовая нотация DRL реализована в виде XML-языка и определена с помощью XML Schema. DRL охватывает все этапы разработки сложной документации — от проектирования до публикации электронных документов. Для удобства публикации текстовая часть DRL интегрирована с известным языком разметки документов DocBook.